INFORMATICA, 2012, Vol. 23, No. 4, 521–536

Learning Process Termination Criteria

Boštjan BRUMEN[1], Marko HÖLBL[1], Katja HAREJ PULKO[1], Tatjana WELZER[1], Marjan HERIČKO[1], Matjaž B. JURIČ[2], Hannu JAAKKOLA[3]

[1] *University of Maribor, Faculty of Electrical Engineering, Computer Science and Informatics Smetanova 17, SI-2000 Maribor, Slovenia*

[2] *University of Ljubljana, Faculty of Computer and Information Science Tržaška cesta 25, SI-1000 Ljubljana, Slovenia*

[3] *Tampere University of Technology Pori, Pohjoisranta 11, FIN-28101 Pori, Finland*

e-mail: marko.holbl@uni-mb.si, bostjan.brumen@uni-mb.si

Received: November 2011; accepted: April 2012

Abstract. In a supervised learning, the relationship between the available data and the performance (what is learnt) is not well understood. How much data to use, or when to stop the learning process, are the key questions.

In the paper, we present an approach for an early assessment of the extracted knowledge (classification models) in the terms of performance (accuracy). The key questions are answered by detecting the point of convergence, i.e., where the classification model's performance does not improve any more even when adding more data items to the learning set. For the learning process termination criteria we developed a set of equations for detection of the convergence that follow the basic principles of the learning curve. The developed solution was evaluated on real datasets. The results of the experiment prove that the solution is well-designed: the learning process stopping criteria are not subjected to local variance and the convergence is detected where it actually has occurred.

Keywords: learning curve, learning process, classification, accuracy, assessment, data mining.

1. Introduction

The supervised learning algorithms have evolved drastically in the last decades. From pure research ideas at the beginning, they are now used in many sophisticated applications, such as autonomous driving of vehicles, design of modern locomotive motors, detecting credit card fraud, assisting in medical diagnostics, and many more.

Induction-based learning algorithms produce models based on the historic data, which are in turn used on new, unseen data. For example, a classification model is built on historic patients' data and this model is then used on a new patient's (measured) data to assist an expert in medical diagnostic process. The performance of the models built from historic data can be measured, such as in the number of correctly classified items compared to the total number of classified items.

The data that is used for building the model is usually abundant. In fact, the amount of data is enormous, and continues to grow at a very fast rate – the amount of data doubles

in approximately 15 months, see, e.g., Moore *et al.* (2007), Hirschler (2010). Building a statistical model using all the available data can be extremely expensive, if not impossible. Additionally, a full multivariable representative sample may not be needed for the purposes of automated learning (Pyle, 1999; Brumen *et al.*, 2007).

The course of data overflow can be tackled in three ways. First, we may speed-up the process by improving the existing algorithms, by developing novel, parallel ones, putting their execution in the clouds, and by implementing machine learning and data mining using parallel approaches; an excellent article about the optimization and knowledge-based techniques is available from Dzemyda and Sakalauskas (2009), an example of such optimization is described by Norkin and Keyzer (2009). Second, we can reduce the vertical dimensionality (i.e., reduce the number of features/attributes) of the data by developing and using the feature selection methods. Third, we can reduce the horizontal dimensionality (i.e., reduce the number of instances) of the data by building the models using not all, but a subset of available data.

In this paper, we deal with the latter. We examine an empirical algorithmic approach to choosing an adequate number of data items for building a model. The number of items is determined by a set of criteria which we develop here.

Paper contribution. The key contribution and the main hypothesis of the paper is as follows: "It is possible to build a set of conditions and confining values at which an additional learning does not contribute to the overall performance of the classification model. The conditions and confining values are based on the properties of a learning curve." Basically, we address the question "When to stop the learning process?" With the answer to this question, we actually find a point of convergence of the learning curve. We developed a set of equations that need to be checked each time the learning process is invoked using the adaptive incremental approach. We prove the hypothesis by designing an experimental study, in which we use empirical and measurable evidence against the developed model.

Paper organization. We present the underlying and related work in Section 2. Here, we give an overview of the related work with the description of a learning curve and how to model it. In Section 3, we outline the basic process for building a learning curve and present a general solution to the problem, i.e., how to observe the learning curve, and develop a set of conditions to be met for the learning process to terminate. In Section 4, we experimentally evaluate various parameters of the solution, and based on the chosen criteria we propose the optimal solution. We conclude the paper with final remarks and comments in Section 5.

2. Related Work

When measuring the accuracy of a classifier (being it a human or a machine), we build a so-called learning curve. The learning curve depicts the relationship between sample size and classifier's performance (see Fig. 1). The horizontal axis represents l, the number of instances in a given training set (l can vary between zero and L, the total number of

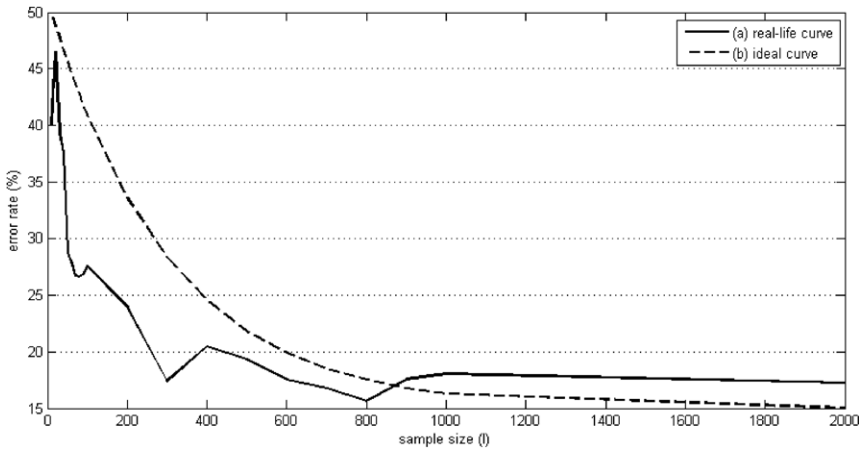

Fig. 1. Real-life (a) and ideal (b) learning curves.

available instances). The vertical axis represents the performance (e.g., error rate) of the model produced by a classifying algorithm when given a training set of size l_i.

Learning curves typically have a steeply sloping portion early in the curve, a more gently sloping middle portion, and a plateau late in the curve. This resembles the way humans learn – Anderson and Schooler (1991) reviewed a number of paradigms in which a power law appears to describe human performance. For this reason the curve is called a learning curve.

The middle portion can be extremely large in some curves and almost entirely missing in others (Harris-Jones and Haines, 1997). The plateau occurs when additional data instances (training) do not improve accuracy. The plateau, and even the entire middle portion, can be missing from curves when l is not sufficiently large. Conversely, the plateau region can constitute the majority of curves when l is very large. The plateau level can be considered as the capacity or the final performance the classifying algorithm (based on the available data).

For example, in their study of two large business data sets, Harris-Jones and Haines (1997) found that learning curves reach a plateau quickly for some algorithms, but small accuracy improvements continue up to L for other algorithms.

Accuracy of a pruned decision tree, generated by C4.5 (Quinlan, 1993) was successfully modeled by the power law by Frey and Fisher (1999). They used power function $y = a \times x^b$ and tried to fit parameters a and b. They also warn that the ability of the power law can also fail if an insufficient amount of data is used for the projection.

Various authors, e.g., Meek *et al.* (2002) and Anderson (2001), confirming our results (Brumen *et al.*, 2001), observe that the typical shape of a learning curve is concave (up) with performance approaching some limiting behavior.

The above mentioned works, and specifically the works of Harris-Jones and Haines (1997), and of Frey and Fisher (1999) lead to the conclusion that the shape of a learning curve (and thus its parameters) depends on both, the type and the contents of the data and the learning algorithm being used.

3. Process of Building and Observing of a Learning Curve

The optimal solution of the learning problem would be a functional dependency between the data, the learning algorithm and its performance (e.g., accuracy). This way we could analytically determine the number of data items needed to build the model for a chosen performance parameter. Unfortunately, for real-life situations, such an approach does not yet exist. It is often so that standard numerical (and other statistical) methods become unstable when using large data sets (Dzemyda and Sakalauskas, 2011). Other theories, such as the founding Vapnik–Chervonenkis theory (Vapnik, 1982), has some limits (e.g., oracle is never wrong) that make it hard to implement in real life situations, as described in detail (Brumen *et al.*, 2007). Analytical results for a specific learner can be found in the literature, e.g., Dučinskas and Stabingiene (2011), but no general ad-hoc analytical solution is available. Thus, we propose an empirical approach where the learning curve is built as early in the learning process as possible and conclusions are drawn from the curve itself. We can do it by empirically measuring the learning curve points and observing the modeled curve on the fly. Algorithm 1 contains the steps for the proposed method of observing a learning curve.

```
01 sample_size:=initial_sample_size;
02 i := 0;
03 difference:=false;
04 REPEAT
05   i := i + 1;
06   class_model := build_classification_model(training_set,
     l_i := training_size);
07   e_i := measure_performance(class_model, testing_set,
     testing_size);
08   IF discrete_concavity_test (l, e) = true THEN
09      e_next := estimate_model_performance( size_next() );
10      e_final := estimate_model_performance( size_final() );
11      difference := estimates_and_actual_performance
        _within_bounds? (e_i, e_next, e_final);
12   END IF
13 increase (training_size, testing_size);
15 UNTIL difference = true OR cancelled;
```

Algorithm 1. Learning curve observation procedure.

A brief description of the procedure is as follows: first, initialize the parameters (Lines 1–3 of the Algorithm 1). Second, build a set of classification models using adaptive incremental k-fold cross-validation (Lines 6–7). Third, after each run, check if the curve meets the discrete concavity criteria (Line 8). If so, estimate the classifier's future performance (Lines 9 and 10), and compare the estimated future performance with the current one. If the estimates and current performance are all within certain bounds (Line 11),

the loop-stop criteria is met, i.e., the future performance of the classifier will not improve. Otherwise, the performance still has some potential, so increase the sample size (Line 13) and loop again (Line 15). The same applies if the discrete concavity criterion is not met. A detailed description and elaboration of each step of Algorithm 1 is as follows:

• *Lines 1 to 5*

First, we initialize the parameters: the initial sample size to be used for building a classification model (sample_size), and the loop variable (difference), and the step number (i), in Lines 1 to 3, respectively. Then the loop is entered (Line 4) and the step number is increased by one (Line 5).

• *Lines 6 and 7*

In Line 6, we build a learning model. After the classification model is built, we can measure its performance (Line 7).

To build a learning curve, though, we repeatedly run a learning algorithm using different sample sizes. We obtain a set of pairs (l_i, e_i), where l_i is a number of data items used for classification model building and e_i is the number of wrongly classified items, i.e., the error rate.

However, single run of the algorithm yields an error rate that is not reliable (Cohen, 1995; Anderson *et al.*, 2008). It does not represent the true error rate: selecting another sample of the same size (especially when the sample size is small) would produce a completely different model and hence a different error rate. Thus, the error rate from a single-run is not statistically valid. In order to get a reliable, statistically valid performance measurement, one could repeatedly train and test a classifier on disjoint sets (each set consisting of randomly drawn items from the total population), then average the scores (i.e., cross-validate) obtained during the testing phases and derive parametric confidence intervals by t-tests, as described by Cohen (1995). Unfortunately, the scheme is very expensive. There is a better way to use training sets. A more efficient training and testing scheme is computer-intensive statistical method called *k-fold cross validation* (Weiss and Kulikowski, 1991). It is efficient in the following sense: we want to train a learning algorithm with all the available data, because its performance is expected to improve as it encounters more items. If we train on l items and immediately test on the same items, however, we will not find out how our algorithm performs on items it has not seen during training. Cross-validation allows us to train a system on almost all the items (90% for ten-fold cross-validation), test it on all the items, and still to discover how the system performs on unseen items. Cross-validation is a resampling technique that reminds of another computer-intensive statistical technique, the bootstrap testing. The common idea that underlies these techniques is to estimate the true (population) performance by treating a sample as if it is the population and repeatedly drawing samples from it. A true bootstrap approach, of course, involves resampling with replacement. The bootstrap procedure thus involves a lot more computation than, say, a ten-fold cross validation. Not

only that it requires much more computation, it is also much more expensive regarding the manual classification of the training and test sample. Bootstrapping namely requires that the sampling is done by the replacement, and this means that once an item has been used (classified), it is returned to the pool of all instances. There is a fair chance that the item will not be used again, while cross-validation technique does not waste the items (no replacement).

The k-fold cross-validation method thus assures that each point of the learning curve (l_i, e_i) is statistically valid and that it truly represents the error rate for any sample of size l_i. The rσ gives a confidence interval for a given sample size (r being the value from the t-test table, σ being calculated from the error rates), which assures that the estimated error rate for that size is statistically valid, meaning that the difference between the calculated (estimated) value and the actual value is insignificant (Brumen *et al.*, 2004). The probability threshold for rejecting the null hypothesis (the measured error rate lies within the confidence interval) was chosen to be $p = 0.001$ because the Bonferroni adjustment requires that the probability of falsely rejecting the null hypothesis need to be adjusted by the number of degrees of freedom (Cohen, 1995), i.e., measurements. The parameter k for k-fold cross-validation was set to 10, as suggested in the literature (Cohen, 1995; Provost *et al.*, 1999; Weiss and Kulikowski, 1991; McLachlan *et al.*, 2004).

Lehnert *et al.* (1993) introduced a variant of a cross validation, called *incremental* k-fold cross-validation, which tests a classifier for a known number of iterations. If the number of iterations is *apriori* unknown, we need to use the *adaptive* incremental k-fold cross-validation (Lines 6 and 7 in Algorithm 1), developed by Brumen *et al.* (2001, 2004). Here, the properties of the learning curve are observed and the procedure is repeated until the loop-until conditions are met. Later, our approach was affirmed to be useful in the works of Castillo and Gama (2006, 2009).

• *Line 8*

The conditions at which to stop the learning process are based on the following basic properties of a learning curve: the error rate is decreasing and the shape of the learning curve is concave up. We define that the learning curve starts "behaving well" when its graph becomes monotonically decreasing and concave up for a given number of points.

The problem with real-life data and algorithms is that the learning curves are not "well behaved". A very important task is to detect the point where the learning curve starts to behave well. Provost *et al.* (1999) define this point as the point of convergence. As mentioned, we do not have any analytical function to observe. Thus, we need to observe the graph of the points that are (later) used to build the analytical model of the learning curve. The observation of the graph that depicts the learning curve is the key address of this paper. We try to answer the question "When to stop the learning process?" In fact, when we answer this question, we find a point of convergence of the learning curve, where the additional learning does not contribute to the overall performance of the classification model.

In the next few paragraphs, we briefly give the definitions for monotonically decreasing (increasing) functions, and concavity of functions, all from Bronshtein *et al.* (2007).

Definition of Monotonicity. Let f be continuous on an open interval, I. The function f is monotonically increasing (decreasing) on I if $f(x_i) \leqslant f(x_j)$ (for decreasing: $f(x_i) \geqslant f(x_j)$) for $x_i < x_j,\ x_i,\ x_j \in I$.

Definition of Concavity. Let f be differentiable on an open interval, I. The graph of f is concave upward on I if *f'* is increasing on the interval and concave downward on I if *f'* is decreasing on the interval.

Concavity Test Theorem. Let f be a function whose second derivative exists on an open interval I. If *f "(x) > 0* for all x in I, then the graph of f is concave upward. If *f "(x) < 0* for all x in I, then the graph of f is concave downward. If $f''(x) = 0$ for all x in I, then the graph of f is a line, neither concave upward or downward. The proof to this theorem is well known and beyond the scope of this paper.

Unfortunately, we only have a set of pairs (l_i, e_i), where l_i is the sample size and e_i is the error rate, and not (yet) a function that would be differentiable. For this reason, we have to modify the definition of concavity to suit our purposes. Analogously to the definition of concavity with continuous functions, we define concavity on the set of discrete points.

Definition of Discrete Concavity. Let f be a set of ordered points (*x, y*) and m be the number of points in the set. The graph of f is concave upward (Fig. 2) if

$$y_{i-2} > y_{i-1} > y_i \tag{1}$$

$$\frac{y_{i-1} - y_{i-2}}{x_{i-1} - x_{i-2}} < \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \tag{2}$$

for $2 \leqslant i \leqslant m$ and $x_{i-2} < x_{i-1} < x_i$. The graph of f is concave down (Fig. 3), if the inequality sign in Eq. (2) is turned.

After building a classifying model using training instances and testing its performance against a test set for a given sample size (Lines 6 and 7), we need to check whether the conditions of discrete concavity are met (Line 8).

• *Lines 9, 10 and 11*

If the conditions of discrete concavity are met, the estimation steps are performed (Lines 9 and 10). In the estimation step, we estimate the error rate for the "next step" (using

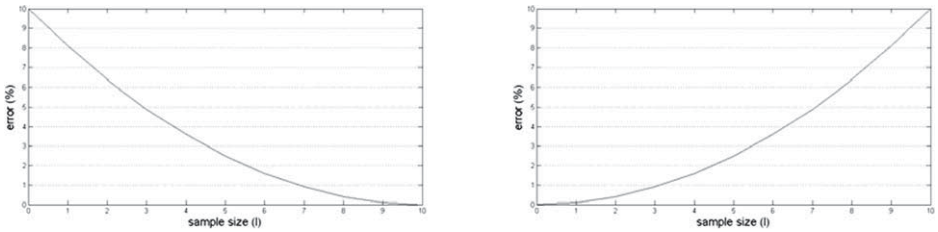

Fig. 2. Concave-up graphs.

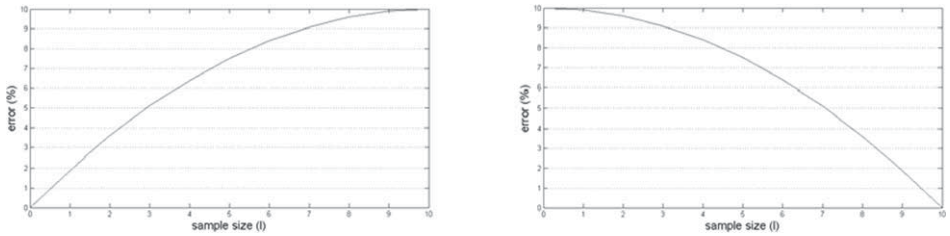

Fig. 3. Concave-down graphs.

$l_{\text{next}} = l_{i+1}$ amount of data) and the "final step" (using l_{large} amount of data). After the estimation is made, the actual error rate, e_i, the estimated error rate for the next step e_{next} and the estimated error rate for a large data size, e_{large} are compared (Line 11). If they are all within a limit (the "ε", provided by the operator), we can conclude that the performance of the algorithm will not improve drastically. To formalize, we check that the Eq. (3) holds:

$$\text{MAX}\big(|e_i - e_{\text{next}}|, |e_{\text{next}} - e_{\text{large}}|, |e_i - e_{\text{large}}|\big) < \varepsilon. \tag{3}$$

This equation prevents from stopping the learning process too soon caused by the local variance, as probably first observed by Provost *et al.*. (1999). The parameter ε depends on the desires and needs of the data analyst. The larger the ε, the sooner the process will stop and the estimates will not be as accurate as the ones calculated with smaller ε. The parameter ε directly influences the number of iterations (the i in Eq. (3). In this paper, we address the selection process of a proper ε as well.

These equations enable the proactive development of the learning curve model: hereby we answer the question "When to stop the learning process?" When the measurements of the learning curve points are finished (based on the properties of the learning curve, of course), the model can be calculated (fitted) from the existing points and the future performance can be assessed. Equation (1) checks the basic property of the learning curve, that is whether it is decreasing or not (for error rate). The increase of the error rate (with the increase of the sample size) indicates anomalies in data and/or in learning algorithm and hints that the learning curve does not behave well yet. Equation (2) checks for the concavity of the graph of the measured points. Here again the behavior of the graph is observed. If it is not concave up (for error rate), this indicates that the decrease is rather sharp and that we are still in the first, steeply portion of the learning curve. Equation (3) checks whether the performance will increase with larger sample size (i.e., the error rate would decrease): if this is so, we can continue with measurements since the improvements are considerable.

• *Lines 12 to 14*

If the conditions of discrete concavity are not met, or the criteria in Eq. (3) are not met, the sample size is increased (Line 13) and the learning process continues, i.e., adaptive k-fold cross-validation loops once again (Line 14).

4. Evaluation of the Convergence Conditions

We evaluated the convergence conditions in five experiments using five data sets, Covertype, Adult, Letter Recognition, Breast Cancer Wisconsin (Original) and Molecular Biology (Splice-Junction Gene Sequences), hereinafter referred to as "Forrest", "Adult", "Letter", "Breast Cancer" and "Genetics", respectively, from UCI Machine Learning Repository (Asuncion and Newman, 2010). The datasets were chosen upon the criteria that the datasets should be public, should be of various sizes and from different domains (and of classification type, of course).

We have used the See5 classification algorithm by Quinlan (2011). The reason we chose this algorithm is that it is freely available for time limited testing and because it is an improved version of the most popular classification algorithm C4.5. C4.5 builds decision trees from a set of training data in the same way as its predecessor ID3 (developed by Quinlan too), using the concept of information entropy. The training data is a set $S = s_1, s_2, \ldots, s_n$ of already classified samples. Each sample $s_i = x_1, x_2, \ldots, x_m$ is a vector where x_j represent attributes or features of the sample. The training data is augmented with a vector $C = c_1, c_2, \ldots, c_n$ where c_i represent the class to which each sample belongs. At each node of the tree, C4.5 chooses one attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. Its criterion is the normalized information gain (difference in entropy) that results from choosing an attribute for splitting the data. The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurses on the smaller sublists (Quinlan, 1993; Kotsiantis, 2007). See5 offers a number of improvements on C4.5, including speed, memory usage, and smaller decision trees. Additionally, it has cross-validation built-in Quinlan (2011). Experimental increments of learning set sizes (l_i) were defined by a sampling schedule $S_a = (100, 200, \ldots, 1000, 2000, \ldots, 10{,}000, 20{,}000, \ldots, 100{,}000, 200{,}000, \ldots)$. The sampling schedule we used is based on efficient progressive adaptive sampling, developed by Provost *et al.* (1999).

We have built a prototype and implemented a method of the adaptive incremental approach as described earlier; for details see Brumen *et al.* (2004). In all the experiments we were using the power law

$$y = a + b \cdot x^c \tag{4}$$

to model the learning curve. For calculation of the parameters a, b and c for power law function, we used Matlab's LSQNONLIN function from the Optimization Toolbox, which is based on the Chi-Squared test. In each step we fitted the parameters to the power law based on the measured performance, i.e., based on a set of measurements (l_i, e_i). When the parameters were calculated, we were able to estimate the future performance of the learning algorithm based on any future sample size; in our case we used it to calculate the expected error rate at the next sample size of the sampling algorithm and the final sample size.

Table 1
List of conditions for detection of convergence

Condition name	Equation name	Equation for condition
C1	Eq. (1)	$y_{i-2} > y_{i-1} > y_i$
C2	Eq. (2)	$\frac{y_{i-1}-y_{i-2}}{x_{i-1}-x_{i-2}} < \frac{y_i-y_{i-1}}{x_i-x_{i-1}}$
C3	Eq. (3) $\mid \varepsilon = 1$	MAX $(\mid e_i - e_{\text{next}} \mid, \mid e_{\text{next}} - e_{\text{large}} \mid, \mid e_i - e_{large} \mid) < 1$
C4	Eq. (3) $\mid \varepsilon = 2$	MAX $(\mid e_i - e_{\text{next}} \mid, \mid e_{\text{next}} - e_{\text{large}} \mid, \mid e_i - e_{\text{large}} \mid) < 2$
C5	Eq. (3) $\mid \varepsilon = 3$	MAX $(\mid e_i - e_{\text{next}} \mid, \mid e_{\text{next}} - e_{\text{large}} \mid, \mid e_i - e_{\text{large}} \mid) < 3$
C6	Eq. (3) $\mid \varepsilon = 4$	MAX $(\mid e_i - e_{\text{next}} \mid, \mid e_{\text{next}} - e_{\text{large}} \mid, \mid e_i - e_{\text{large}} \mid) < 4$
C7	Eq. (1) $\wedge$ Eq. (2)	$y_{i-2} > y_{i-1} > y_i \wedge \frac{y_{i-1}-y_{i-2}}{x_{i-1}-x_{i-2}} < \frac{y_i-y_{i-1}}{x_i-x_{i-1}}$
C8	Eq. (1) $\wedge$ Eq. (2) $\wedge$ Eq. (3) $\mid \varepsilon = 1$	$\text{MAX}(\mid e_i - e_{\text{next}} \mid, \mid e_{\text{next}} - e_{\text{large}} \mid, \mid e_i - e_{\text{large}} \mid) < 1 \wedge y_{i-2} \wedge > y_{i-1} > y_i \wedge \frac{y_{i-1}-y_{i-2}}{x_{i-1}-x_{i-2}} < \frac{y_i-y_{i-1}}{x_i-x_{i-1}}$
C9	Eq. (1) $\wedge$ Eq. (2) $\wedge$ Eq. (3) $\mid \varepsilon = 2$	$\text{MAX}(\mid e_i - e_{\text{next}} \mid, \mid e_{\text{next}} - e_{\text{large}} \mid, \mid e_i - e_{\text{large}} \mid) < 2 \wedge y_{i-2} > y_{i-1} > y_i \wedge \frac{y_{i-1}-y_{i-2}}{x_{i-1}-x_{i-2}} < \frac{y_i-y_{i-1}}{x_i-x_{i-1}}$
C10	Eq. (1) $\wedge$ Eq. (2) $\wedge$ Eq. (3) $\mid \varepsilon = 3$	$\text{MAX}(\mid e_i - e_{\text{next}} \mid, \mid e_{\text{next}} - e_{\text{large}} \mid, \mid e_i - e_{\text{large}} \mid) < 3 \wedge y_{i-2} > y_{i-1} > y_i \wedge \frac{y_{i-1}-y_{i-2}}{x_{i-1}-x_{i-2}} < \frac{y_i-y_{i-1}}{x_i-x_{i-1}}$
C11	Eq. (1) $\wedge$ Eq. (2) $\wedge$ Eq. (3) $\mid \varepsilon = 4$	$\text{MAX}(\mid e_i - e_{\text{next}} \mid, \mid e_{\text{next}} - e_{\text{large}} \mid, \mid e_i - e_{\text{large}} \mid) < 4 \wedge y_{i-2} > y_{i-1} > y_i \wedge \frac{y_{i-1}-y_{i-2}}{x_{i-1}-x_{i-2}} < \frac{y_i-y_{i-1}}{x_i-x_{i-1}}$

In this section, however, we analyze and evaluate the convergence conditions of the Eqs. (1), (2), and (3), the latter for different ε, and a selected combination thereof. The conditions are presented in the Table 1.

To evaluate each condition, we measured the differences between the estimated error rate and the measured error rate in the step i, where each of the conditions first evaluated to true. We calculated the differences between the estimations and the real (measured) values. We made an estimation of error rate in the next step $(i+1)$ and of error rate in the final step (i_f), because Eq. (3) so requires.

Please note that the differences between the estimated and the actual values can only be calculated *post festum*, after the learning process (i.e., building a classifier for a given sample size) has completed, not *a priori* at the learning time. Thus, this analysis and the comparison of the conditions can not be performed prior or while the learning process is running, but only after it has completed for a given sample size.

The differences are presented in Tables 2 and 3. Smallest difference is emphasized. Where the corresponding condition did not evaluate to true, the table has an empty cell, meaning the convergence was not detected.

The comparison of the conditions based on the difference between the estimated final and the measured final error rate has shown that the best condition is C9 (Eqs. (1), (2) and (3) with $\varepsilon = 2\%$) followed by C8 (Eq. (3) with $\varepsilon = 2\%$) and C10. However, the

Table 2
Differences between the estimated and the measured error rate in the final step

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
Adult	2.43	8.66	0.39	0.33	0.34	**0.27**	2.38	0.36	0.39	0.39	**0.27**
Letter	3.49	39.69					**3.49**				
Covtype	13.27	5.67	15.5	15.4	15.4	15.4	3.27		**0.4**	3.28	3.28
Breast-Cancer	0.61	0.51	0.41	**0.31**	0.61	0.61	0.51	0.7	0.51	0.51	0.51
Genetics	7.09	3.84	**0.42**	**0.42**	3.11	3.48	3.84	**0.42**	**0.42**	**0.42**	3.48
Average	*5.38*	*11.67*	*4.18*	*4.11*	*4.86*	*4.94*	*2.69*	*0.49*	***0.43***	*1.15*	*1.88*

Table 3
Differences between the estimated and the measured error rate in the next step

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
Adult	1.16	1.44	0.38	0.47	**0.10**	1.55	1.77	0.81	0.15	0.15	1.55
Letter	**3.76**	15.91					**3.76**				
Covtype	0.52	4.29	2.51	0.98	0.98	0.98	2.73		**0.35**	2.73	2.73
Breast-Cancer	2.12	0.25	**0.10**	0.25	0.25	2.12	0.25	0.39	0.25	0.25	0.25
Genetics	10.83	1.98	**1.11**	**1.11**	1.95	2.21	1.98	**1.11**	**1.11**	**1.11**	2.21
Average	*3.68*	*4.77*	*1.02*	*0.70*	*0.82*	*1.71*	*2.10*	*0.77*	***0.47***	*1.06*	*1.69*

condition C8 has failed at the Covtype (Forrest) dataset. The best condition C9 has an average difference, across all the datasets, between the estimated and the measured final error rate of 0.43%. The second best average difference (0.49%) is C8, but this condition has failed to discover the convergence for Covtype dataset.

The comparison of the conditions based on the difference between the estimated and the measured next error rate has shown that the best condition is C9, followed by C4 and C8, as can be seen in Fig. 4 (please note the logarithmic scale of Y axis).

The best condition C9 has an average difference, across all the datasets, between the estimated and the measured next error rate of 0.47%. The second best average difference (0.70%) is produced at the condition C4, followed by C8 (0.77%).

To double check whether C9 is the best condition, we also check the ranks of the conditions. We do so since the average differences can be skewed when one condition is performing particularly badly with a single dataset (e.g., C2 with Letter dataset) while performing well on others. Rank one (the first) is assigned to the smallest difference between the estimated and the actual value (values in bold in Table 2) and the comparison of the conditions based on the difference between the estimated final and the measured final error rate has shown that the best condition is C9 (Eqs. (1), (2), and (3) with $\varepsilon = 2\%$) followed by C8 (Eq. (3) with $\varepsilon = 2\%$) and C10. However, the condition C8 has failed at the Covtype (Forrest) dataset. The best condition C9 has an average difference, across all the datasets, between the estimated and the measured final error rate of 0.43%. The second best average difference (0.49%) is C8, but this condition has failed to discover the

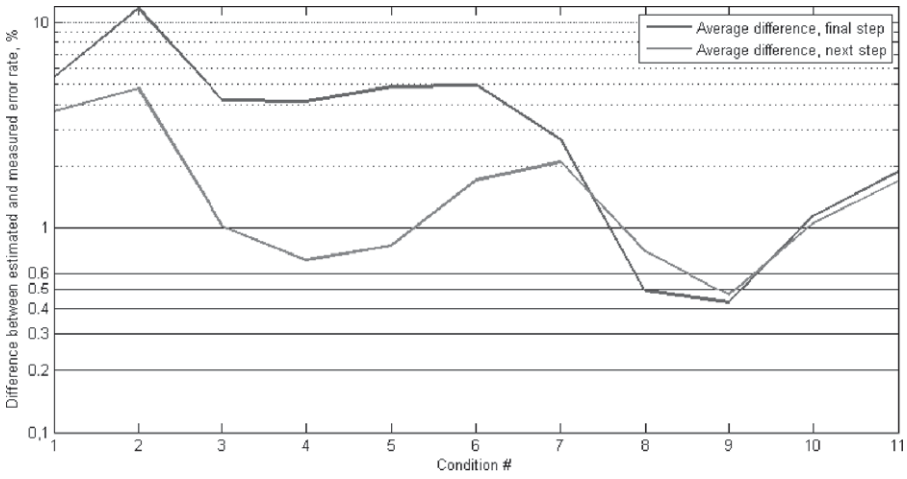

Fig. 4. Average differences between estimated and measured error rate at different conditions.

Table 4
Ranking of the conditions on "final error rate"

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
Adult	10	11	6	3	4	1	9	5	6	6	1
Letter	11	11	11	11	11	11	11	11	11	11	11
Covtype	6	5	10	7	7	7	2	11	1	3	3
Breast-Cancer	8	3	2	1	8	8	3	11	3	3	3
Genetics	11	9	1	1	6	7	9	1	1	1	7
Average with Letter	*9.2*	*7.8*	*6*	*4.6*	*7.2*	*6.8*	*6.8*	*7.8*	***4.4***	*4.8*	*5*
Average w/o Letter	*8.75*	*7*	*4.75*	*3*	*6.25*	*5.75*	*5.75*	*7*	***2.75***	*3.25*	*3.5*

convergence for Covtype dataset (Table 3).

If two differences are the same, they get assigned the same rank, and the next difference gets assigned rank $+2$. Where the convergence was not detected, the condition gets assigned rank no. 11 (out of 11 conditions). With Letter dataset, the convergence did not occur, see Brumen (2004). Thus, also the conditions that discovered the convergence (that actually did not occur) get assigned rank 11. Ranking of the conditions on "final" and "next" error rate are presented in Tables 4 and 5, respectively, with presented average rank including and excluding the Letter dataset. Average of the average rank is presented in Table 6.

The comparison of the conditions has shown that by using both criteria the condition C9 has lowest average rank, followed by C4, C10 and C3.

The average ranks of different conditions are presented in Fig. 5.

The average difference and the average ranking have both shown that the condition C9 is best performing. Furthermore, only the C9 discovers the convergence where it happens and overcomes the problem of local variance of error rate. Other conditions fail at

Table 5
Ranking of the conditions on "next error rate", sorted

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
Adult	7	8	4	5	1	9	11	6	2	2	9
Letter	11	11	11	11	11	11	11	11	11	11	11
Covtype	2	10	6	3	3	3	7	11	1	7	7
Breast-Cancer	10	2	1	2	2	10	2	9	2	2	2
Genetics	11	7	1	1	6	9	7	1	1	1	9
Average with Letter	*8.2*	*7.6*	*4.6*	*4.4*	*4.6*	*8.4*	*7.6*	*7.6*	***3.4***	*4.6*	*7.6*
Average w/o Letter	*7.5*	*6.75*	*3*	*2.75*	*3*	*7.75*	*6.75*	*6.75*	***1.5***	*3*	*6.75*

Table 6
Average ranking of the conditions on both tests, sorted

	C9	C4	C10	C3	C5	C11	C7	C6	C2	C8	C1
Average with Letter	*3.9*	*4.5*	*4.7*	*5.3*	*5.9*	*6.3*	*7.2*	*7.6*	*7.7*	*7.7*	*8.7*
Average w/o Letter	*2.12*	*2.87*	*3.12*	*3.87*	*4.62*	*5.12*	*6.25*	*6.75*	*6.87*	*6.87*	*8.12*

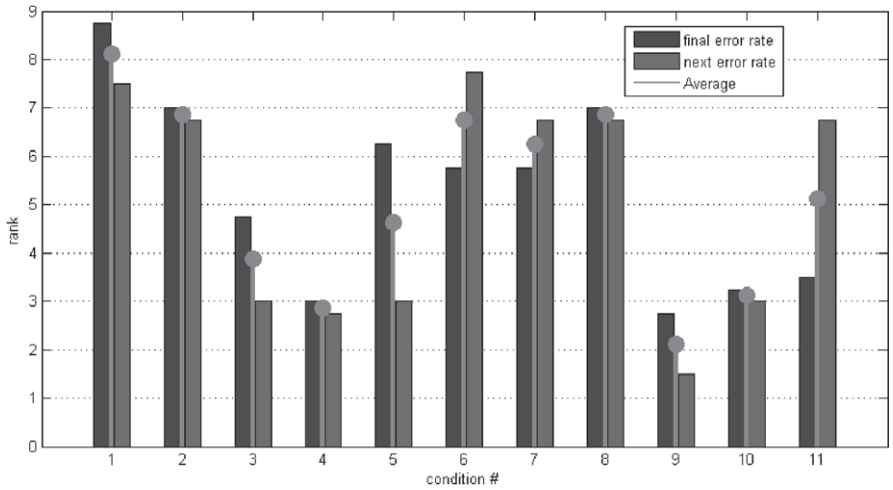

Fig. 5. Average ranks across conditions.

solving either of the problem of local variance or at discovering the convergence. Thus, we propose the condition C9 (Eqs. (1), (2), and (3) with $\varepsilon = 2\%$) as optimal for ending the learning process and detecting the convergence; this condition answers the question "When to stop measuring the learning curve points?" best.

5. Conclusion

In this paper we presented a solution to the question "When to stop the learning process?" By using the developed equations, we can find the point of convergence of the learning curve. At this point the additional learning does not contribute to the overall performance of the classification model any more. We developed a set of equations that need to be checked after the learning process using a given number of data items in the learning set has finished. If the conditions are not satisfied, the algorithm has a potential to learn more. The developed solution has proven to be working properly: the learning process is not subjected to local variance and the convergence is detected where it actually occurred.

The contribution of this work can be incorporated into the algorithms building classification models. The user could get an indication whether any additional data is needed to improve the performance, or contrary, no more data is needed since the model is performing adequately well.

There are many areas for future investigation. For example, one can consider (1) other aspects of the learning curve beyond the concavity and the decrease of the error rate, (2) other factors of the learning process, such as algorithm's run time or associated costs, or (3) other cases of learning, not only classification. Another interesting area of the investigation is human learning, where our model can be – in principle – applied to the individuals in the learning process. Here, the individuals and their knowledge could be evaluated and the convergence of the learning process could be detected – a point where additional learning would not yield any considerable improvement.

References

Anderson, J.R., Schooler, L.J. (1991). Reflections of the environment in memory. *Psychological Science*, 2, 396–408.

Anderson, R.B. (2001). The power law as an emergent property. *Memory & Cognition*, 29, 1061–1068.

Anderson, R.B., Doherty, M.E., Friedrich, J.C. (2008). Sample size and correlational inference. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 34, 929.

Asuncion, A., Newman, D. (2010). *UCI Machine Learning Repository* [Online]. University of California, School of Information and Computer Science, Irvine. Available: `http://archive.ics.uci.edu/ml/datasets.html`. Accessed 2011-11-11.

Bronshtein, I.N., Semendiaev, K.A., Musiol, G., Muehlig, H. (2007). *Handbook of Mathematics*. Springer, New York.

Brumen, B. (2004). *Empirical procedure for assessment of classification algorithms.* Doctor of Science thesis, University of Maribor.

Brumen, B., Jaakkola, H., Welzer, T. (2001). Predicting sample size in data mining tasks using additive incremental approach. *Information Modelling and Knowledge Bases XII*, 67, 264.

Brumen, B., Golob, I., Jaakkola, H., Welzer, T., Rozman, I. (2004). Early assessment of classification performance. In: Hogan, J., Montague, P., Purvis, M., Steketee, C. (Eds.), *ACSW Frontiers*, Australian Computer Society, Inc., Dunedin, pp. 91–96.

Brumen, B., Juric, M.B., Welzer, T., Rozman, I., Jaakkola, H., Papadopoulos, A. (2007). Assessment of classification models with small amounts of data. *Informatica*, 18, 343–362.

Castillo, G., Gama, J. (2006). An adaptive prequential learning framework for Bayesian network classifiers. In: *Knowledge Discovery in Databases (PKDD 2006)*, pp. 67–78.

Castillo, G., Gama, J. (2009). Adaptive Bayesian network classifiers. *Intelligent Data Analysis*, 13, 39–59.

Cohen, P.R. (1995). *Empirical Methods for Artificial Intelligence*. MIT Press, Cambridge.

Dučinskas, K., Stabingiene, L. (2011). Expected bayes error rate in supervised classification of spatial Gaussian data. *Informatica*, 22, 371–381.

Dzemyda, G., Sakalauskas, L. (2009). Optimization and knowledge-based technologies. *Informatica*, 20, 165–172.

Dzemyda, G., Sakalauskas, L. (2011). Large-scale data analysis using heuristic methods. *Informatica*, 22, 1–10.

Frey, L.J., Fisher, D.H. (1999). Modeling decision tree performance with the power law. In: Heckerman, D., Whittaker, J. (Eds.), *Seventh International Workshop on Artificial Intelligence and Statistics*. Kaufmann, San Francisco.

Harris-Jones, C., Haines, T.L. (1997). Sample size and misclassification: Is more always better? AMS Center for Advanced Technologies.

Hirschler, B. (2010). Roche fears drug industry drowning in "spam" data. *Reuters* [Online]. Available: `http://www.reuters.com/article/2010/12/01/roche-data-idUSLDE6B01XF20101201` [Accessed 2011-11-11].

Kotsiantis, S.B. (2007). Supervised machine learning: a review of classification techniques. *Informatica*, 31, 249–268.

Lehnert, W., Mccarthy, J., Soderland, S., Riloff, E., Cardie, C., Peterson, J., Feng, F. F., Dolan, C., Goldman, S. (1993). UMass/Hughes: description of the CIRCUS system used for MUC-5. In: *Fifth Message Understanding Conference*. Association for Computational Linguistics/Kaufmann, pp. 277–291.

Mclachlan, G.J., Do, K.-A., Ambroise, C. (2004). *Analyzing Microarray Gene Expression Data*. Wiley, Hoboken.

Meek, C., Thiesson, B., Heckerman, D. (2002). The learning-curve sampling method applied to model-based clustering. *The Journal of Machine Learning Research*, 2, 397–418.

Moore, R.L., D'aoust, J., Mcdonald, R.H., Minor, D. (2007). Disk and tape storage cost models. *IS&T Archiving*. Society for Imaging Science and Technology, Arlington.

Norkin, V., Keyzer, M. (2009). On stochastic optimization and statistical learning in reproducing kernel Hilbert spaces by support vector machines (SVM). *Informatica*, 20, 273–292.

Provost, F., Jensen, D., Oates, T. (1999). Efficient progressive sampling. In: Chaudhuri, S., Madigan, D. (Eds.), *Fifth International Conference on Knowledge Discovery and Data Mining*, ACM, San Diego, pp. 23–32.

Pyle, D. (1999). *Data Preparation for Data Mining*. Kaufmann, San Francisco.

Quinlan, R.J. (1993). *C4. 5: Programs for Machine Learning*. Kaufmann, San Francisco.

Quinlan, R.J. (2011). *Data Mining Tools See5 and C5.0* [Online]. Available: `http://www.rulequest.com/`. Accessed 2011-11-11.

Vapnik, V.N. (1982). *Estimation of Dependences Based on Empirical Data*. Springer, New York.

Weiss, S.M., Kulikowski, C.A. (1991). *Computer Systems That Learn: Classification and Prediction Methods from Statistics, Neural Nets, Machine Learning, and Expert Systems*. Kaufmann, San Mateo.

B. Brumen received doctor's degree in informatics in 2004. He is an assistant professor at University of Maribor, Faculty of Electrical Engineering and Computer Science. He was secretary general (provost) of University of Maribor for two consecutive terms between 2004 and 2011. His research interests include intelligent data analysis, automated learning, data security and data quality.

K. Harej Pulko received her PhD (informatics) in 2009 from University of Maribor. Her research interests cover open source systems, project management, virtual communities and knowledge management.

M. Hölbl is an assistant professor and researcher at the Data Technology Laboratory, Faculty of Electrical Engineering and Computer Science, University of Maribor. His research interests include computer and information security, cryptography, eLearning and databases.

T. Welzer is a full time professor and head of the Data Technology Laboratory, Institute of Informatics at the Faculty of Electrical Engineering and Computer Science, University of Maribor. Her research interests include computer security, cultural and human factors of computer security, databases and medical informatics.

M. Heričko Heričko is a professor at the Faculty of Electrical Engineering and Computer Science, University of Maribor. He received his PhD in 1998 from the University of Maribor. His research interests include all aspects of information system development with emphasis on design patterns, service-oriented architectures and software metrics.

M. B. Jurič is a professor at the Faculty of Computer and Information Science at the University of Ljubljana. He has authored several books and articles, and received awards such as Java Champion, IBM Champion and Oracle ACE Director. His research area covers cloud computing, service oriented architectures and integration of information systems.

H. Jaakkola is a professor of software engineering at Tampere University of Technology and director of Pori Doctor School. He received his PhD in engineering at Tampere University of Technology in 1991. His research interests include software engineering methods, software quality, data mining, data management, technology management, technology diffusion and technology transfer.

Mokymosi proceso sustabdymo kriterijai

Boštjan BRUMEN, Marko HÖLBL, Katja HAREJ PULKO, Tatjana WELZER,
Marjan HERIČKO, Matjaž B. JURIČ, Hannu JAAKKOLA

Prižiūrimo mokymosi santykis tarp turimų duomenų ir klasifikavimo kokybės nėra pakankamai gerai suprantamas. Kiek duomenų naudoti, arba, kada sustabdyti mokymosi procesą, yra svarbiausi klausimai. Šiame straipsnyje, pristatome metodą, leidžiantį preliminariai vertinti išgautų žinių kokybę ir rezultatų tikslumą. Atsakymas į pagrindinius klausimus yra randamas aptinkant pagrindinį konvergavimo tašką. T.y. kai klasifikavimo kokybė nebegerėja nors ir pridedant papildomų mokymosi duomenų. Kad nustatyti mokymosi proceso baigties momentą mes sukūrėme keletą lygčių kurios paremtos pagrindiniais mokymosi proceso principais. Sukurtas sprendimas buvo vertinamas naudojant realaus pasaulio duomenis. Eksperimento rezultatai parodo, kad sprendimas yra gerai parengtas: mokymosi procesas sustojimo kriterijai nėra priklausomas nuo lokalios dispersijos ir konvergencija yra aptinkama ten kur ir iš tikrųjų turėtų būti.